无迹西格玛点引导的拟反向黏菌算法及其工程应用 *

刘宇凇，刘　升†

(上海工程技术大学 管理学院，上海 201600)

摘　要：针对黏菌算法搜索停滞和算法稳定性差等问题，提出了无迹西格玛点引导的拟反向黏菌算法。首先，使用了拟反向学习和拟反射学习两种反向学习过程，根据原始黏菌算法勘探和开采行为的表现时机，生成同时包含拟反向和拟反射的综合反向种群，扩大搜索范围；其次，根据种群的多样性程度判断是否使用反向种群重构原始种群进行后续计算，避免固定的反向过程破坏种群本身的搜索特点，提高搜索精度；最后，引入无迹变换的西格玛点，改进黏菌算法的基本移动模式，使无迹西格玛点引导黏菌算法的搜索，加快收敛速度。实验部分使用了 CEC2017 基准测试函数，使用了传统统计特征和 MAE 排名、Wilcoxon 秩和检验验证算法有效性，并使用其求解轿车侧面碰撞的实际工程优化问题，与新颖的高水平群智能算法、改进算法、不完全算法进行对比测试，实验结果表明提出的改进策略有效且各策略间组合相得益彰，改进后算法的求解精度和鲁棒性更具竞争力。

关键词：黏菌算法；拟反向学习；拟反射学习；无迹变换；CEC2017

中图分类号：TP301.6　　doi: 10.19734/j.issn.1001-3695.2022.02.0055

Unscented sigma point guided quasi-opposite slime mould algorithm and its application in engineering problem

Liu Yusong, Liu Sheng†

(*School of Management Studies*, *Shanghai University of Engineering Science*, *Shanghai* 201600, *China*)

Abstract: Aiming at the search stagnation and poor stability of the Slime Mould algorithm, propose an unscented sigma point guided quasi-opposite Slime Mould Algorithm. Firstly, use quasi-opposite learning and quasi-reflected learning to exploration and exploitation behaviors according to the original Slime Mould Algorithm to generate a comprehensive opposite population that includes both quasi-opposite learning and quasi-reflected learning, expand the search space. Secondly, according to the diversity of the population, decide whether to use the opposite population to regenerate a new population for subsequent calculations, avoid the continuous opposite process destroying the search characteristics of the population itself, and improve the search accuracy. Finally, use unscented transformation Sigma point to improve the basic movement mode of Slime Mould Algorithm, make the unscented sigma point guide the search, and accelerate the convergence speed. The experimental part uses the CEC2017 benchmark test functions, uses traditional statistical index and MAE ranking, Wilcoxon rank-sum test to verify the effectiveness of the algorithm, and use it to solve the car side impact design problem, compares and tests with the novel swarm intelligence algorithms, improved algorithms and incomplete algorithm. The experimental results show that the strategies is effective and combinations of strategies complement each other, and the improved algorithm's solution accuracy and robustness are more competitive.

Key words: slime mould algorithm; quasi-opposite learning; quasi-reflected learning; unscented transformation; CEC2017

0　引言

在大数据和人工智能的背景下，群智能启发式算法因其在大型复杂优化问题中优秀的表现和低廉的算力要求在近十几年间愈发流行。黏菌算法[1](slime mould algorithm，SMA)是 2020 年 Shimin Li 等人提出的一个用于求解随机优化问题的算法，其本身的移动原理简单，但蕴涵有动态自适应、反馈权重等改进机制，同时具有广泛的扩展定制空间，受到了国外研究者广泛的关注。黏菌算法由自然中黏菌的振荡模式启发而设计，通过生物振荡来传递不同位置食物信息的正向或负向反馈，从而寻找到最优的食物位置。目前该算法已成功应用于光伏模型参数优化[2]、机器学习特征筛选[3]、医学 X 光图像分割[4]、计算机视觉图像阈值[5]等实际问题，在金融科技、数据挖掘、工业工程等领域具有广阔的发展前景。

然而黏菌算法依旧存在着搜索停滞、稳定性差、求解精度低等问题，虽然现阶段对 SMA 算法的研究相对较少，国内外学者依旧使用不同的策略进行改进获得了不同的成果。在使用反向学习框架改进黏菌算法的方向上，文献[6]使用混沌反向学习框架扩大算法搜索空间；文献[7]使用随机反向学习框架帮助算法跳出局部最优；文献[8]使用精英反向和二次插值的综合策略改进了黏菌算法，使其整体表现更优。在改进黏菌算法基本移动方式的方向上，文献[3]使用名为"觅食-进攻"的策略，基于贪婪策略和高斯变异，针对单个个体位置每轮迭代中进行新位置的尝试，使得求解精度大大提高；文献[9]将人工蜂群的交叉算子引入黏菌算法，加快算法收敛速度；文献[10]将正余弦算法的搜索方式添加进黏菌算法，提高算法的搜索效率；文献[11]使用量子旋转门替换黏菌算法的基础移动方式，改善了算法搜索停滞、陷入局部最优的缺陷。在使用反向学习框架改进黏菌算法的研究中，其大多使用单一反向策略，且对反向种群重构的时机缺乏判断和控制，在

收稿日期：2022-02-24；**修回日期：**2022-04-17　**基金项目：**国家自然科学基金资助项目(61673258，61075115)；上海市自然科学基金资助项目(19ZR1421600)

作者简介：刘宇凇(1997-)，男，内蒙古包头人，硕士，主要研究方向为智能算法；刘升(1966-)，男(通信作者)，湖北黄石人，教授，硕导，博士，主要研究方向为智能计算、群智能系统、进化算法(ls6601@sina.com).

保持算法原本特征和减少算法时间复杂度上存在进一步优化空间；在改进黏菌算法基本移动方式的研究中，其主要使用经典改进策略和其他种类算法的核心移动方式替换黏菌算法移动方式，对黏菌算法不同搜索行为的阶段缺乏区分和针对性调整，在保持算法原有优势的基础上弥补算法局限性上存在进一步优化空间。

针对黏菌算法搜索停滞、稳定性差的问题，本文使用反向学习的策略框架改进黏菌算法，并在传统反向学习的基础上针对黏菌算法自身特点进行策略的调整和组合，提出了无迹西格玛点引导的拟反向黏菌算法(UQSMA)。使用拟反向和拟反射的综合反向过程跳出局部最优，保持种群多样性；同时根据种群多样性适时使用反向种群重构原始种群，减少固定反向过程的运算开销，加强算法的稳定性；使用无迹西格玛点引导种群向理论最优逼近，针对黏菌算法不同的搜索阶段使用不同的改进策略，弥补算法在面对不同实验和不同问题时稳定性较差的局限。多种策略平衡了种群的勘探和开采阶段，各策略组合后在保证单一策略有效的基础上弥补了单一策略的局限。实验中，测试了 CEC2017 基准测试函数，实验结果验证了改进策略对算法求解精度和鲁棒性的积极影响。

1　基本黏菌算法

黏菌算法中的黏菌一般是指多头绒泡菌，因其最初被归类为真菌，所以被命名为“黏菌”。黏菌是一种生活在寒冷潮湿地方的真核生物，黏菌的活跃动态阶段是黏菌算法研究的主要阶段。在这个阶段，黏菌使用体内的有机物寻找食物，包围它，并分泌酶来消化它。在黏菌迁移过程中，其前端会延伸成扇形，后端则连接着一个静脉网络，保证细胞质在其中流动，从而达到信息的沟通，并且根据搜寻到食物的质量和密度动态调整静脉直径的大小，最终将整个黏菌的位置调整至最优食物处。受到黏菌这种行为模式的启发，Shimin Li 及其团队设计开发了黏菌算法。

基本黏菌算法的抽象数学描述如下：假设 n 个黏菌分布在 d 维搜索空间中，第 t 次迭代第 i 个黏菌的位置可以表示为 $X_i(t)$，$\max_t$ 为最大迭代次数，全局最优位置 X_b 表示食物位置。黏菌的觅食寻优过程分为三个阶段：

1) 接近食物

黏菌可以根据空气中的气味接近食物，下式模拟了其接近食物的过程：

$$X(t+1)=\begin{cases}X_b(t)+vb\cdot(W\cdot X_A(t)-X_B(t)),r<p\\ vc\cdot X(t),r\geq p\end{cases} \tag{1}$$

其中，vb 是服从 $[-a,a]$ 均匀分布的系数，vc 从 0 到 1 线性递增，t 代表当前迭代轮次，X_b 代表目前为止找到食物浓度最高的个体的位置，即当前全局最优解，X 代表黏菌位置，X_A 和 X_b 代表两个从黏菌种群中随机抽取的个体，W 代表黏菌个体的权重。

计算 p 的公式如下：

$$p=\tanh|S(i)-DF| \tag{2}$$

其中 $i\in1,2,\ldots,n$，$S(i)$ 表示黏菌个体的适应度值，DF 表示所有迭代轮次中找到的最优适应度。

计算 vb 的公式如下：

$$vb=[-a,a] \tag{3}$$

其中 a 为随迭代次数减小的动态参数，用来收缩随机数 vb 的生成范围，以此模拟黏菌接近食物源过程中静脉逐渐收缩提高寻优精度的过程，a 的计算公式如下：

$$a=arctanh\left(-\left(\frac{t}{\max_t}\right)+1\right) \tag{4}$$

计算权重 W 的公式如下：

$$W(SmellIndex(i))=\begin{cases}1+r\cdot\log\left(\frac{bF-S(i)}{bF-wF}+1\right),condition\\ 1-r\cdot\log\left(\frac{bF-S(i)}{bF-wF}+1\right),others\end{cases} \tag{5}$$

$$SmellIndex=sort(S) \tag{6}$$

其中选择分支的判断条件 $condition$ 为适应度排名位于前一半种群，r 为 $[0,1]$ 间的一个随机数，$\max_t$ 表示最大迭代次数，bF 表示当前轮次获得的最优适应度值，相对的，wF 表示当前轮次获得的最差适应度值，$SmellIndex$ 表示按照适应度值大小排序的黏菌个体索引(在求解最小值问题时使用升序排序方法)。

2) 缠绕食物

该阶段模拟了黏菌搜索食物过程中静脉结构的收缩模式，当黏菌静脉接触到的食物浓度越大，其通过生物振荡反应产生的波就越强，细胞质流动就越快，静脉细胞质粘稠程度也就越高。式(5)描述了黏菌在搜索区域基于食物浓度或质量的判断获得的反馈，若该位置的食物浓度处于整个种群截至目前所有搜索结果的上位，则给予该位置黏菌正向反馈，反之给予下位黏菌反向反馈，以此方式充分利用适应度的计算结果信息来调整不同位置黏菌的权重，进一步调整整个种群的搜索方式。同时，随机数 r 保证了黏菌个体间的多样性。

基于上述理论，黏菌的位置更新公式如下：

$$X^*=\begin{cases}rand\cdot(UB-LB)+LB,rand<z\\ X_b(t)+vb\cdot(W\cdot X_A(t)-X_B(t)),r<p\\ vc\cdot X(t),r\geq p\end{cases} \tag{7}$$

其中 LB 和 UB 代表了搜索空间的上下边界约束，$rand$ 和 r 代表了 $[0,1]$ 间随机数。转换概率 z 在 Shimin Li 的参数实验中被证设置为 0.03 时整个算法表现最好，故本文中 z 同样设置为固定值 0.03。

3) 振荡反应

黏菌依靠生物的震荡反映来调整静脉中细胞质的流动从而调整自身位置搜索最优食物位置。黏菌算法中的参数 W、vb、vc 用来模拟黏菌的振荡反应，W 根据适应度选择不同振荡模式的行为模拟了黏菌个体根据其他位置个体搜索到的食物情况选择自身下一步行为的搜索模式，vb 动态调整振荡过程中使用其他个体信息的口径模拟了黏菌静脉直径变化的过程，vc 线性变化模拟了黏菌对自身历史信息保留程度的变化。

2　无迹西格玛点引导的拟反向黏菌算法

2.1　反向学习策略

Tizhoosh[13] 首次提出了反向学习(opposition-based learning，OBL)的概念，其广泛应用于群智能算法的改进策略中，同时反向学习也具有许多扩展和变体形态。反向学习策略有助于扩大搜索空间，加速算法收敛，维持种群多样性。反向学习的核心机制是同时评估原始种群和反向种群的适应度，并从中挑选适应度更优的个体组成新的种群参与后续计算。传统反向点生成的数学表达式如下：

$$x_j^o=LB_j+UB_j-x_j \tag{8}$$

拟反向学习[14](quasi-opposition-based learning，QOBL)和拟反射学习[15](quasi-reflection-based learning，QRBL)是反向学习的两种变体，拟反向和拟反射点生成的数学表达如下：

$$x_j^{qo}=rand\left[\left(\frac{LB_j+UB_j}{2}\right),(LB_j+UB_j-x_j)\right] \tag{9}$$

$$x_j^{qr}=rand\left[\left(\frac{LB_j+UB_j}{2}\right),x_j\right] \tag{10}$$

文献[16]使用了哈里斯鹰算法(HHO)中控制种群实施不同搜索行为的参数控制通过拟反向和拟反射过程生成的反向种群。证明了拟反向学习过程可以规避传统反向学习位置转

换过大导致的精度不高问题，同时可以相对地扩大搜索空间加速种群收敛；拟反射学习可以扩大个体位置附近的搜索空间，提高种群的搜索精度。图 1 展示了一维空间内拟反向学习和拟反射学习相对于传统反向学习的搜索空间。

图 1　三种反向学习在一维空间的搜索空间示意图

Fig. 1　Search space for three OBL in 1-dimensional space

标准的群智能算法框架中存在着明确的勘探和开采行为的区分，若使用单一的反向学习策略，生成的反向种群特征也较为单一，对算法勘探和开采阶段的辅助效果会不同，且会存在一个效果相对不好的阶段。为了更有效地构建反向种群，本文在黏菌算法进行勘探时使用大步长粗糙搜索的拟反向学习过程生成反向个体，在开采时使用小步长精细搜索的拟反射学习过程生成反向个体，最终生成完整的综合反向种群，提升反向学习框架针对不同搜索空间特征的优化效果。在黏菌算法中，参数 p 控制着种群的勘探和开采行为，本文在黏菌进行勘探时使用拟反向过程生成反向个体，在年黏菌进行开采时使用拟反射过程生成反向个体，数学表达式如下：

$$OX=\begin{cases} rand\left[\left(\frac{LB+UB}{2}\right),(LB+UB-x)\right],r<p \\ rand\left[\left(\frac{LB+UB}{2}\right),x\right],r\geq p \end{cases} \tag{11}$$

其中参数 p 随着迭代轮次的增加而减小，即生成的综合反向种群在算法前期以拟反向个体为主，在算法后期以拟反射个体为主，从而动态平衡反向种群内个体勘探和开采的数量比例。

2.2　线性多样性度量

文献[17]提出了度量进化算法中种群多样性的度量方式，其数学表达式如下：

$$D_d(P_t)=\frac{1}{D}\sqrt{\sum_{d=1}^{D}\overline{(x_t^d)^2}-\left(\overline{x_t^d}\right)^2} \tag{12}$$

其中：

$$\overline{(x_t^d)^2}=\frac{1}{NP}\sum_{i=1}^{NP}(x_{i,t}^d)^2 \qquad \overline{x_t^d}=\frac{1}{NP}\sum_{i=1}^{NP}x_{i,t}^d \tag{13}$$

其中，D 为问题维度，d 为当前维度，NP 为种群规模。

传统反向学习中，每轮迭代生成的反向种群都会与原始种群一起进行比较从而生成新种群，不仅会大大提升运算时间开销，新种群位置的重置还会打断原始种群的收敛过程，导致算法在多次独立实验时表现出不稳定的问题，因此适时地进行反向重构种群才能够最大程度的发挥反向学习策略的优势。

受文献[19]启发，本文设置一个种群多样性阈值(diversity threshold，DT)用以调控当前种群是否需要通过上述综合反向种群生成新种群。若计算得出的种群多样性大于阈值，则生成新种群；若种群多样性小于阈值，则不生成新种群。由于黏菌算法本身具有维持自身种群多样性的能力，故当种群多样性变小时可能表明种群处于收敛过程。在实验中发现，即使黏菌算法陷入局部最优，其种群多样性并不会始终维持低水平，而是会在高低水平间震荡波动，此时若种群多样性超过多样性阈值，综合反向种群便可以通过生成新种群帮助原始种群扩大搜索范围，跳出局部最优。该机制以较小的计算量达到了缩减反向重构过程的效果，显著的减少了运算的时间开销；同时由于种群多样性与种群的收敛与否具有较强相关性，因此通过设定阈值对种群状态进行筛选后重构得到的新种群也一定程度上可以对应种群的不同收敛状态，从而在种群陷入停滞时进行帮助，在种群快速收敛时不进行干扰。

2.3　无迹西格玛点引导策略

文献[20]首次提出了使用无迹变换(unscented transformation，UT)生成一系列无迹西格玛点帮助原始种群逼近理论最优点位置。通过修正原始算法的步进方式，丰富种群搜索的方式，提高算法面对不同问题时的泛用性，避免算法陷入停滞，无迹西格玛点数学表达式如下：

$$\begin{cases} \xi_0' = \overline{x} \\ \xi_d' = \overline{x_d}+\sigma_d, d=1,2,\ldots,D \\ \xi_{d+D}' = \overline{x_d}-\sigma_d, d=1,2,\ldots,D \end{cases} \tag{14}$$

其中 $\overline{x_d}$ 为维度 d 上位置的平均值，σ_d 为维度 d 上位置的标准差。

由上式可知无迹西格玛点共 $2D+1$ 个。本文使用第一个无迹西格玛点，即种群位置的均值来引导黏菌算法的勘探阶段；使用剩余无迹西格玛点，即使用标准差和均值运算得出的新位置来引导黏菌算法的开采阶段，帮助算法在反向学习扩展后的搜索空间中引导算法逼近理论最优。式(7)为黏菌算法的核心搜索框架，其中 $r<p$ 时种群表现出勘探的行为，其基本移动方式为使用最优位置部分信息的基础上使用两个随机个体位置信息生成步长，本文将 X_A 替换为 X，减少步长生成的随机性，使用均值引导种群在勘探时快速收敛；$r\geq p$ 时种群表现出开采的行为，其基本移动方式为对自身位置的衰减，本文将其替换为均值与方差生成的新位置，同时保留参数 vc 保证一定的随机性，帮助黏菌算法在开采阶段具有逼近理论最优点的能力，同时设定转换概率，不全部计算 $2D$ 个无迹西格玛点，节省运算开销。本文使用无迹西格玛点改进后的式(7)如下：

$$X^*=\begin{cases} rand\cdot(UB-LB)+LB, rand<z \\ X_b(t)+vb\cdot\left(W\cdot\overline{\mathrm{X}(t)}-X_B(t)\right), r<p \\ \overline{\mathrm{X}(t)}+vc\cdot\sigma(t), r\geq p \quad r'>0.5 \\ \overline{\mathrm{X}(t)}-vc\cdot\sigma(t), r\geq p \quad r'\leq 0.5 \end{cases} \tag{15}$$

2.4　无迹西格玛点引导的拟反向黏菌算法

本文提出了无迹西格玛点引导的拟反向黏菌算法(unscented sigma point guided quasi-opposite Slime Mould Algorithm，UQSMA)。首先，根据原始黏菌算法勘探和开采的不同阶段，使用拟反向学习和拟反射学习过程生成综合反向种群，在黏菌种群不同的空间特征搜索阶段使用不同的反向过程有助于放大黏菌在不同搜索行为时的倾向，强化黏菌算法本身的行为特征，加速算法的启动；其次，使用线性多样性度量计算种群多样性，并根据种群的实时多样性判断是否使用综合反向种群加入原始种群生成新种群，保证算法精度，节约运算开销，提高算法的稳定性；最后，使用无迹西格玛点替换原始黏菌算法的基本移动方式，引导种群逼近理论最优解，丰富搜索的路径，使算法保持最优解的更新和优化，避免算法停滞。

2.5　改进算法的步骤法

UQSMA 算法的流程图如图 2 所示。

2.6　时间复杂度分析

假设种群规模为 NP，搜索空间维度为 D，最大迭代次数为 T，原始 SMA 算法的时间复杂度主要来源为初始化、适应度计算和位置更新，其时间复杂度为

$$O(SMA)=O(NP)+O(NP*T)+O(NP*D*T) \tag{16}$$

UQSMA 算法主要由拟反向和拟反射学习组成的综合反向学习框架和无迹西格玛点引导机制两个改进策略组成。反向过程需要根据 NP 个个体生成 NP 个反向个体，增加计算 NP 个反向个体的适应度，对所有 $2NP$ 个个体进行排序并生成新种群，该过程时间复杂度为 $O(NP*D+2NP+2NPlg2NP)$，与

标准反向学习算法相同；无迹西格玛点生成过程不增加时间复杂度，则 UQSMA 算法的时间复杂度如下：

$$\begin{aligned}&O(UQSMA)=O(NP+(NP*D+2NP+2NP\lg 2NP))\\&+O(NP*T)+O((2NP*D+2NP+2NP\lg 2NP)*T)\\&=O(NP*(3+D+2\lg 2NP+T(3+2D+2\lg 2NP)))\end{aligned}\tag{17}$$

UQSMA 算法实际运行时，由于存在适应度阈值判断是否进行综合反向过程，故根据实际问题的不同算法总时间复杂性有不同程度的降低，即 $O((2NP*D+2NP+2NP\lg 2NP)*T)$ 部分在实际运算过程中会更小，故式(17)是 UQSMA 算法时间复杂度的理论最大值。

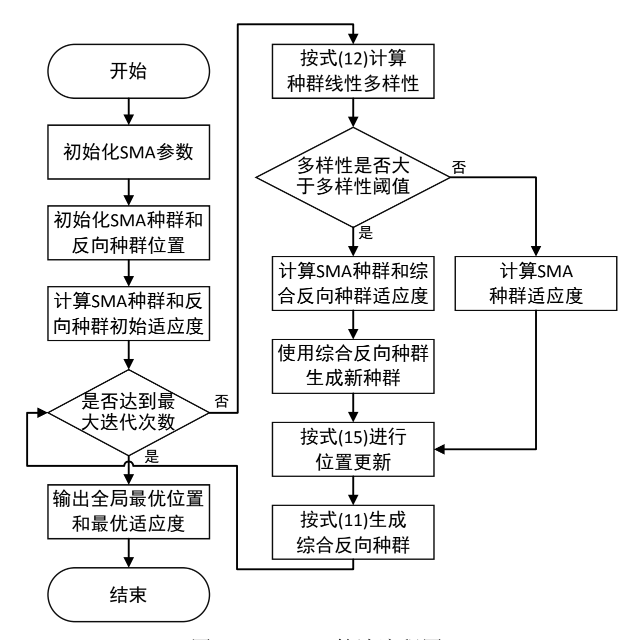


图 2　UQSMA 算法流程图

Fig. 2　Flow chart of UQSMA

2.7　收敛性分析

UQSMA 算法本质是一种随机搜索算法，可由概率测度法证明 UQSMA 的全局收敛性。全局收敛性准则及定理[18]阐述了算法满足全局收敛性需要满足的两个条件：

条件 1　$f(D(z,\xi))\le f(z)$，且若 $\xi\in S$，则

$$f(D(z,\xi))\le f(\xi)$$

其中，f 为适应度函数，D 为随机搜索算法，z 为解空间 R^N 子集 S 的其中一个点，该点能使适应度函数取得最小值或在 S 上产生可接受的函数值下确界，ξ 为算法 D 在迭代搜索过程中得到的解。

条件 2　对于 S 中任意的 Borel 子集 A，若其概率测度 $v[A]>0$，则有

$$\prod_{k=0}^{\infty}(1-\mu_k(A))=0$$

其中，$\mu_k(A)=P(x^k\in A|x^0,x^1,\ldots x^{k-1})$ 为算法 D 第 k 次迭代结果在集合 A 上的概率测度，$\{x_k\}_{k=1}^{\infty}$。该假设表明，对于 S 中任意满足 $v[A]>0$ 的子集 A 经过算法的无穷次迭代后，错过解空间 S 的任意的 Borel 子集 A 的概率为 0，即满足上述条件的算法在无穷次搜索后无法达到近似全局最优点的概率为 0。

引理 1　随机优化算法全局收敛定理[18].若函数 f 可测，搜索空间 S 是 R^n 上的可测子集，若算法满足条件 1 和条件 2，则有

$$\lim_{k\to\infty}P[x^k\in R]=1$$

其中，R 是全局最优点集合，$P[x^k\in R]$ 是算法第 k 次搜索迭代结果落在 R 中的概率。

定理 1　UQSMA 满足条件 1。

证明　本文算法最优位置更新包含原始黏菌算法的位置更新和拟反向与拟反射学习的位置更新，故可将算法 D 抽象定义为

$$D(G_t,X_{i,t})=\begin{cases}G_t, & f(g(X_{i,t}))\ge f(G_t)\\ g(X_{i,t}), & f(g(X_{i,t}))<f(G_t)\end{cases}$$

和

$$D(G_t,X_{i,t})=\begin{cases}G_t, & f(h(X_{i,t}))\ge f(G_t)\\ h(X_{i,t}), & f(h(X_{i,t}))<f(G_t)\end{cases}$$

其中，g 对应拟反向与拟反射学习过程，与标准反向学习过程在位置更新步骤上逻辑一致，$X_{i,t}$ 表示 R^N 中第 t 代个体 i 的样本空间，$g(X_{i,t})$ 表示第 t 次迭代中通过拟反向和拟反射学习生成的综合反向种群与原始黏菌种群整合后更新的个体 i 位置，$h(X_{i,t})$ 表示黏菌个体 i 经过位置移动后第 t 次更新后的位置，G_t 为当前得到的全局最优位置。由上述表达式可知，G_t 所对应的全局最优适应度单调不增，且逐渐向解空间的下确界收敛。

证明成立。

定理 2　UQSMA 满足条件 2。

证明　若算法满足条件 2，种群规模为 NP 的黏菌种群的样本空间必须包含 S，即

$$S\subseteq\bigcup_{i=1}^{N}M_{i,t}$$

其中，$M_{i,t}$ 为第 t 次迭代中个体 i 的样本空间支撑集。

对于 UQSMA 中拟反向和拟反射学习过程的位置更新机制，有

$$g(X_{i,t})=\begin{cases}rand\left[\left(\dfrac{LB+UB}{2}\right),(LB+UB-X_{i,t})\right],r<p\\ rand\left[\left(\dfrac{LB+UB}{2}\right),X_{i,t}\right],r\ge p\end{cases}$$

由此可知 $g(X_{i,t})$ 为 R^N 中不同于 $X_{i,t}$ 的样本空间，即经过拟反向和拟反射学习过程后 $X_{i,t}$ 到达 $g(X_{i,t})$ 的概率大于零。因此随着迭代次数的增加每个 $g(X_{i,t})$ 的闭包 $v[g(X_{i,t})]$ 与其并集 $\bigcup_{i=1}^{N}g(X_{i,t})$ 的闭包 $v\left[\bigcup_{i=1}^{N}g(X_{i,t})\right]$ 都在逐渐扩大，因此存在一个整数 t_1，当 $t>t_1$ 时

$$S\subseteq\bigcup_{i=1}^{N}g(X_{i,t})$$

同理，原始黏菌算法的位置更新过程得到的样本空间并集的闭包 $v\left[\bigcup_{i=1}^{N}h(X_{i,t})\right]$ 也在逐渐扩大。因此存在一个整数 t'，当 $t>t'$ 时

$$\left(\bigcup_{i=1}^{N}h(X_{i,t})\right)\cup\left(\bigcup_{i=1}^{N}g(X_{i,t})\right)\supseteq S$$

令 S 的 Borel 子集 $D=\left(\bigcup_{i=1}^{N}h(X_{i,t})\right)\cup\left(\bigcup_{i=1}^{N}g(X_{i,t})\right)$，有 $v[D]>0$，$\mu_t(D)=\sum_{i=1}^{N}\mu_{i,t}(D)=1$，即

$$\prod_{k=0}^{\infty}(1-\mu_t[D])=0$$

证明成立。

综上，由引理 1 得，UQSMA 具有全局收敛性，且由于拟反向和拟反射学习过程生成的新样本空间不同于原始黏菌种群的样本空间，该策略可以扩大整个种群在一次迭代中搜索的样本空间，故收敛速度相对于原始黏菌算法有一定加快。

3　仿真实验及结果分析

实验环境为 Windows10，64 位操作系统，CPU 为 AMD Ryzen 7 5800H，主频 3.20GHz，内存 16GB，算法基于 MATLAB2020b 编写。

实验一　为了充分验证本文所提出的 UQSMA 在求解函数优化问题时的求解精度和稳定性，将 UQSMA 算法与其他

算法在 CEC2017 测试函数上进行求解，并进行实验结果的对比。对比用算法包括最新的改进黏菌算法 AFSMA[3]、动态反向改进的黏菌算法 DOLSMA、精英反向与其他策略结合改进的鲸鱼优化算法 EGSWOA[22]、原始的黏菌算法 SMA[1]、同一科研团队提出的哈里斯鹰算法 HHO[23]和经典的鲸鱼优化算法 WOA[21]。

为保证每种算法对比时的公平性，实验中种群规模设置为 30，最大迭代次数设置为 500，并独立运行 30 次计算得出统计结果。对比用算法参数设置均来源于对应文献设置，如表 1 所示。

本文选用了表 2 所列示的 29 个 CEC2017 基准测试函数，其类型包含单峰(Unimodal, UM)、多峰(Multimodal, MM)、混合(Hybrid, H)及复合(Composition, C)，维度可选用 10、30、50、100，此处实验使用标准维度 30 维，02 号函数由于高维的不稳定性习惯上不使用。

表 3 为 UQSMA 与第一部分算法进行对比的实验结果。总体上看，UQSMA 在 29 个 CEC2017 基准测试函数中在 21 个函数上平均求解精度的表现优于其他所有对比算法，说明本文改进策略能够使算法保证持续的搜索效果和搜索效率，克服原始黏菌算法搜索停止的局限。在此基础上 UQSMA 算法的求解精度均高于传统反向改进的黏菌算法 DOLSMA，说明拟反向和拟反射学习的组合对于黏菌算法效果的提升优于传统的动态反向学习。同时，UQSMA 算法的方差整体低于其他算法，说明了本文策略的组合对于算法稳定性的提升是有效的，对于算法最终求解精度和鲁棒性具有一定的效果。

表 1　主要参数

Tab. 1　Main parameters

算法	主要参数
UQSMA	z=0.03 ; DT=0.3
AFSMA	σ=1
DOLSMA	z=0.03
EGSWOA	b=1 ; x1=-π+(1-τ)*2π ; x2=-π+τ*2π
SMA	z=0.03
HHO	β=1.5
WOA	b=1

表 2　CEC2017 函数

Tab. 2　CEC2017 function

函数	函数名称	类型	最优值	函数	函数名称	类型	最优值
CEC01	Shifted and Rotated Bent Cigar Function	UM	100	CEC16	Hybrid Function 5 (N=4)	H	1600
CEC03	Shifted and Rotated Zakharov Function	UM	300	CEC17	Hybrid Function 6 (N=4)	H	1700
CEC04	Shifted and Rotated Rosenbrock’s Function	MM	400	CEC18	Hybrid Function 6 (N=5)	H	1800
CEC05	Shifted and Rotated Rastrigin’s Function	MM	500	CEC19	Hybrid Function 6 (N=5)	H	1900
CEC06	Shifted and Rotated Expanded Scaffer’s F6 Function	MM	600	CEC20	Hybrid Function 6 (N=6)	H	2000
CEC07	Shifted and Rotated Lunacek Bi-Rastrigin Function	MM	700	CEC21	Composition Function 1 (N=3)	C	2100
CEC08	Shifted and Rotated Non-Continuous Rastrigin’s Function	MM	800	CEC22	Composition Function 2 (N=3)	C	2200
CEC09	Shifted and Rotated Lévy Function	MM	900	CEC23	Composition Function 3 (N=4)	C	2300
CEC10	Shifted and Rotated Schwefel’s Function	MM	1000	CEC24	Composition Function 4 (N=4)	C	2400
CEC11	Hybrid Function 1 (N=3)	H	1100	CEC25	Composition Function 5 (N=5)	C	2500
CEC12	Hybrid Function 2 (N=3)	H	1200	CEC26	Composition Function 6 (N=5)	C	2600
CEC13	Hybrid Function 3 (N=3)	H	1300	CEC27	Composition Function 7 (N=6)	C	2700
CEC14	Hybrid Function 4 (N=4)	H	1400	CEC28	Composition Function 8 (N=6)	C	2800
CEC15	Hybrid Function 5 (N=4)	H	1500	CEC29	Composition Function 9 (N=3)	C	2900

表 3　CEC2017 测试函数和 Wilcoxon 秩和检验的结果对比

Tab. 3　Comparison of CEC2017 function results and Wilcoxon’s rank-sum test

	UQSMA	AFSMA	DOLSMA	EGSWOA	SMA	HHO	WOA
	Mean(Std)win	Mean(Std)win	Mean(Std)win	Mean(Std)win	Mean(Std)win	Mean(Std)win	Mean(Std)win
CEC01	**1.21E+04**(8.85E+03)	4.33E+04(1.07E+05)=	5.50E+05(3.24E+05)+	2.90E+10(6.55E+09)+	1.19E+05(4.92E+04)+	3.81E+08(2.39E+08)+	5.65E+09(2.26E+09)+
CEC03	**1.26E+04**(5.16E+03)	4.80E+04(1.44E+04)=	3.27E+04(1.16E+04)=	8.39E+04(6.63E+03)=	3.61E+04(1.60E+04)=	5.88E+04(6.44E+03)=	2.80E+05(6.03E+04)=
CEC04	**5.01E+02**(2.58E+01)	5.11E+02(2.75E+01)+	5.03E+02(1.54E+01)+	6.34E+03(1.50E+03)+	5.20E+02(3.98E+01)+	7.20E+02(9.25E+01)+	1.38E+03(3.85E+02)+
CEC05	**6.04E+02**(3.07E+01)	6.74E+02(3.36E+01)=	6.37E+02(3.44E+01)=	8.99E+02(4.09E+01)+	6.29E+02(3.93E+01)=	7.69E+02(3.14E+01)+	8.59E+02(5.33E+01)+
CEC06	6.19E+02(7.99E+00)	6.24E+02(1.02E+01)+	6.21E+02(1.16E+01)+	6.83E+02(1.21E+01)+	**6.18E+02**(1.14E+01)-	6.68E+02(5.38E+00)+	6.84E+02(1.31E+01)+
CEC07	**8.51E+02**(2.78E+01)	9.35E+02(5.40E+01)+	8.93E+02(4.03E+01)=	1.38E+03(6.75E+01)+	8.91E+02(4.95E+01)=	1.32E+03(6.42E+01)+	1.32E+03(7.53E+01)+
CEC08	**9.00E+02**(1.96E+01)	9.54E+02(3.33E+01)+	9.31E+02(3.10E+01)+	1.12E+03(2.63E+01)+	9.34E+02(3.67E+01)+	9.93E+02(2.96E+01)+	1.07E+03(4.85E+01)+
CEC09	**1.99E+03**(6.54E+02)	5.05E+03(1.56E+03)+	4.39E+03(1.60E+03)+	1.08E+04(1.32E+03)+	4.37E+03(1.47E+03)+	8.92E+03(8.65E+02)+	1.06E+04(3.37E+03)+
CEC10	**4.79E+03**(6.36E+02)	5.24E+03(7.21E+02)+	5.05E+03(8.10E+02)+	8.39E+03(6.40E+02)+	4.97E+03(6.99E+02)+	6.09E+03(8.64E+02)+	7.37E+03(6.18E+02)+
CEC11	1.26E+03(4.50E+01)	**1.25E+03**(4.85E+01)-	1.28E+03(5.74E+01)=	7.12E+03(1.88E+03)+	1.29E+03(6.31E+01)=	1.54E+03(1.70E+02)+	1.02E+04(4.98E+03)+
CEC12	3.56E+06(2.84E+06)	**3.37E+06**(3.13E+06)=	5.58E+06(4.07E+06)=	2.02E+09(8.57E+08)+	4.74E+06(3.82E+06)=	9.00E+07(1.01E+08)+	4.58E+08(2.46E+08)+
CEC13	6.82E+04(3.64E+04)	**3.78E+04**(6.34E+04)=	1.01E+05(5.07E+04)+	4.07E+08(3.97E+08)+	6.60E+04(3.35E+04)=	9.18E+06(3.98E+07)+	1.47E+07(2.11E+07)+
CEC14	**4.07E+04**(3.79E+04)	5.05E+05(4.93E+05)+	1.79E+05(1.92E+05)+	2.36E+06(1.61E+06)+	2.10E+05(2.20E+05)=	1.40E+06(1.62E+06)+	3.11E+06(2.72E+06)+
CEC15	4.58E+04(2.43E+04)	**1.33E+04**(1.33E+04)-	2.78E+04(1.98E+04)+	7.88E+07(8.33E+07)+	2.47E+04(1.64E+04)+	1.36E+05(9.08E+04)+	9.31E+06(1.94E+07)+
CEC16	2.61E+03(2.28E+02)	2.64E+03(3.03E+02)+	2.61E+03(3.02E+02)+	4.62E+03(5.33E+02)+	**2.56E+03**(3.47E+02)-	3.70E+03(5.09E+02)+	4.28E+03(5.92E+02)+
CEC17	**2.13E+03**(1.75E+02)	2.45E+03(2.64E+02)=	2.36E+03(2.37E+02)=	2.92E+03(3.67E+02)+	2.38E+03(2.18E+02)=	2.71E+03(3.45E+02)+	2.91E+03(2.84E+02)+
CEC18	**1.02E+06**(7.54E+05)	2.45E+06(2.72E+06)+	2.28E+06(3.35E+06)+	1.84E+07(1.82E+07)+	2.20E+06(2.35E+06)+	2.40E+06(1.92E+06)+	1.33E+07(1.27E+07)+
CEC19	2.45E+04(2.25E+04)	**1.19E+04**(1.56E+04)-	2.14E+04(1.97E+04)=	9.63E+07(9.51E+07)+	1.77E+04(1.91E+04)+	2.09E+06(1.70E+06)+	1.58E+07(1.31E+07)+
CEC20	**2.53E+03**(1.56E+02)	2.71E+03(1.88E+02)+	2.60E+03(1.76E+02)=	2.90E+03(1.85E+02)+	2.65E+03(1.98E+02)=	2.81E+03(1.87E+02)+	2.92E+03(2.15E+02)+
CEC21	**2.39E+03**(4.08E+01)	2.45E+03(3.67E+01)+	2.43E+03(3.34E+01)=	2.67E+03(5.12E+01)+	2.44E+03(3.85E+01)+	2.60E+03(5.07E+01)+	2.66E+03(5.08E+01)+
CEC22	**3.05E+03**(1.53E+03)	6.13E+03(1.85E+03)+	5.82E+03(1.74E+03)+	7.88E+03(1.26E+03)+	6.26E+03(9.23E+02)+	7.57E+03(1.01E+03)+	7.82E+03(1.84E+03)+
CEC23	**2.76E+03**(2.58E+01)	2.80E+03(3.96E+01)+	2.77E+03(2.60E+01)+	3.24E+03(1.30E+02)+	2.78E+03(3.14E+01)+	3.32E+03(1.35E+02)+	3.16E+03(9.95E+01)+
CEC24	**2.94E+03**(4.54E+01)	2.98E+03(4.81E+01)+	2.96E+03(3.77E+01)=	3.44E+03(1.16E+02)+	2.96E+03(3.56E+01)+	3.50E+03(1.80E+02)+	3.30E+03(9.78E+01)+
CEC25	**2.90E+03**(1.71E+01)	2.91E+03(2.90E+01)+	2.90E+03(1.44E+01)=	3.85E+03(1.72E+02)+	2.91E+03(2.34E+01)=	3.02E+03(3.77E+01)+	3.21E+03(8.34E+01)+
CEC26	**4.35E+03**(9.63E+02)	5.37E+03(4.12E+02)=	4.83E+03(4.63E+02)=	9.24E+03(7.12E+02)+	5.02E+03(5.90E+02)=	8.17E+03(8.36E+02)+	8.59E+03(1.14E+03)+
CEC27	3.24E+03(1.95E+01)	3.25E+03(2.09E+01)+	3.24E+03(1.61E+01)=	**3.20E+03**(1.80E-09)-	3.24E+03(1.77E+01)+	3.69E+03(1.90E+02)+	3.51E+03(1.51E+02)+
CEC28	**3.24E+03**(2.49E+01)	3.27E+03(2.65E+01)=	3.27E+03(3.93E+01)=	3.30E+03(4.00E-10)+	3.27E+03(4.50E+01)=	3.49E+03(1.24E+02)+	3.97E+03(4.12E+02)+
CEC29	**3.95E+03**(2.02E+02)	4.02E+03(2.94E+02)+	4.06E+03(2.25E+02)+	5.87E+03(7.31E+02)+	4.04E+03(2.38E+02)+	4.97E+03(5.31E+02)+	5.34E+03(5.61E+02)+
CEC30	4.28E+05(2.27E+05)	**3.28E+04**(1.43E+04)=	2.56E+05(2.14E+05)=	2.11E+08(1.30E+08)+	1.17E+05(9.64E+04)=	2.46E+07(4.10E+07)+	7.72E+07(5.38E+07)+
+/=/-		17/9/3	14/15/0	27/1/1	14/13/2	28/1/0	28/1/0

为了进一步验证改进算法的有效性，需要对群智能算法结果进行统计检验[24]，本文选择 Wilcoxon 秩和检验进行统计检验，显著性水平设置为 5%，表 3 展示了 UQSMA 算法与其他算法的检验结果，每列最后一个符号为 UQSMA 算法与算法成对比较结果的胜负标记，其中"+"代表均值结果优于比较算法且 Wilcoxon 秩和检验证明两算法结果分布具有差异，即 UQSMA 算法综合优于对比算法；"="代表无论均值优劣且 Wilcoxon 秩和检验证明两算法结果分布不具有明显差异，即 UQSMA 算法综合与对比算法相似；"-"代表均值结果劣于比较算法且 Wilcoxon 秩和检验证明两算法结果分布具有差异，即 UQSMA 算法综合劣于对比算法。最后一行为优于、相似与劣于的函数数量合计。表 3 结果表明 UQSMA 算法对于每个 CEC2017 基准测试函数的求解结果完全不劣于 DOLSMA、HHO、WOA 三个对比算法；相较于改进算法 EGSWOA 仅在 1 个测试函数处于劣势和在 1 个测试函数上表现相似，其他函数均处于优势；相较于 AFSMA 算法和原始 SMA 算法，UQSMA 算法仅在少数测试函数上处于劣势。相较于原始 SMA，接近半数的测试函数结果分布无明显差异，分析可知 UQSMA 算法通过设定种群多样性阈值适时的进行反向过程重构原种群，保留了原始 SMA 的搜索特征，但在此基础上最终的求解结果多数优于原始 SMA 算法，说明本文策略间组合可以达到在不改变原始 SMA 算法搜索特征的基础上帮助算法持续搜索，防止算法停滞，从而提高最终的算法求解精度，可以证明 UQSMA 的持续搜索效果在求解 CEC2017 基准测试函数上表现更优。

为了形象的观察改进算法的收敛情况与稳定性，绘制 UQSMA 与上述对比算法的收敛曲线和箱线图。图 3 给出部分基准测试函数的收敛曲线图,各算法曲线图例与图 3(h)一致，为了便于观察高精度的曲线收敛情况，纵坐标取以 10 为底的对数绘制所有曲线图像。图 4 给出部分基准测试函数带均值的箱线图。观察可得，UQSMA 算法具有前期收敛快、后期保持持续搜索的优势，说明了拟反向过程和拟反射过程在前中期扩大了搜索空间，从而使算法快速的找到更优的寻优方向从而加速收敛；无迹西格玛点对于种群搜索行为的修正帮助算法在迭代后期依旧能够保持持续的精进，避免算法搜索停滞。同时，UQSMA 在保证精度的情况下，重复计算间的方差更小，稳定性更强，说明反向学习扩大搜索空间的特点有助于保持算法的稳定，即使反向过程增加了种群的多样性、无迹西格玛点引入了新的随机性，组合后算法得益于根据多样性进行反向重构的策略依旧具有优于对比算法的稳定性。直观来看，本文改进策略的组合对防止算法停滞和加强算法稳定性有积极影响。

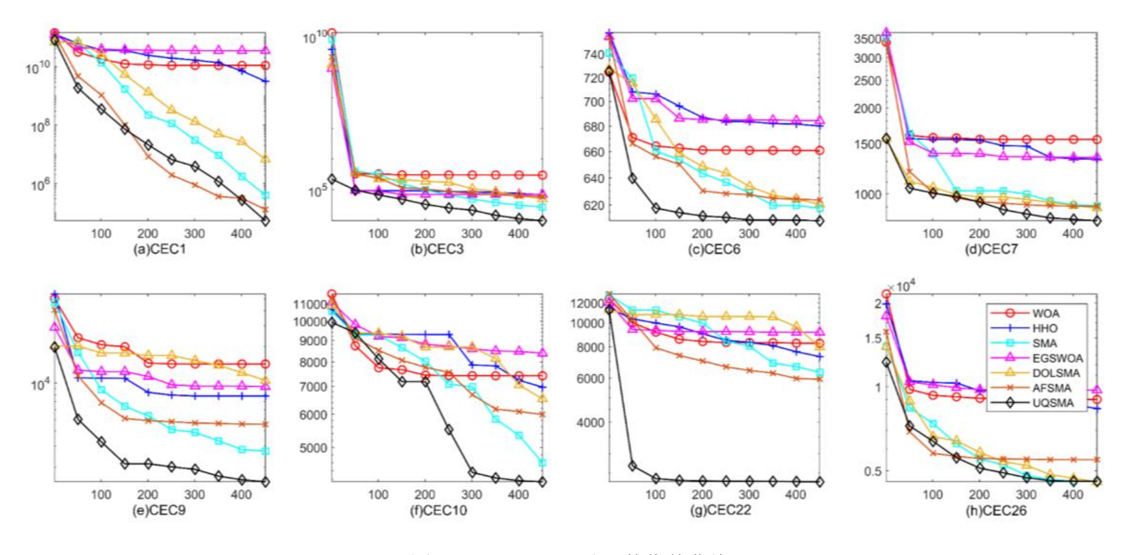


图 3　CEC2017 测试函数收敛曲线

Fig. 3　CEC2017 Function average convergence curve

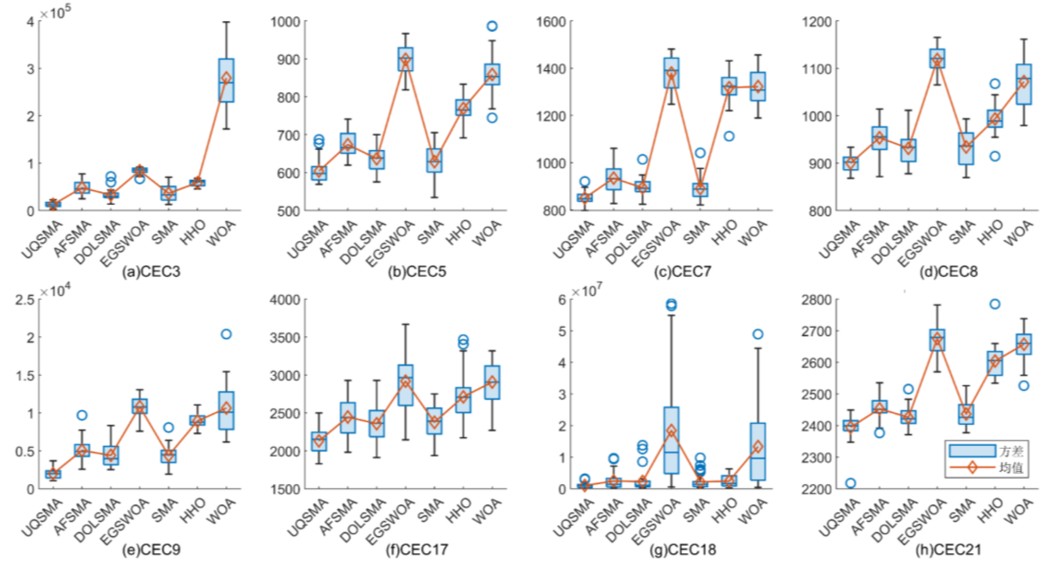


图 4　CEC2017 测试函数箱线图

Fig. 4　CEC2017 Function box chart

为了综合、简明地评估 UQSMA 与其他算法对比的竞争性，使用平均绝对误差 MAE 对上述算法进行排序[25]。表 4 展示了所有算法基于 29 个 CEC2017 基准测试函数的 MAE。MAE 计算公式如下：

$$MAE = \frac{1}{NF}\sum_{i=1}^{NF}\left|f_i - f_i^*\right| \tag{18}$$

其中 NF 为函数个数，f_i 为算法在第 i 个函数上求解结果的平均值，f_i^* 为第 i 个函数的理论最优值。

表 4　MAE 算法排名

Tab. 4　MAE algorithm ranking

算法	MAE	RANK
UQSMA	1.80E+05	1
AFSMA	2.25E+05	2
SMA	2.60E+05	3
DOLSMA	3.12E+05	4
HHO	1.76E+07	5
WOA	2.15E+08	6
EGSWOA	1.10E+09	7

由表 4 可知，UQSMA 排名第一，并且 UQSMA 的 MAE 值明显小于其他算法，更加综合的表明了该算法的有效性和鲁棒性。

实验二　为了验证 UQSMA 算法改进的有效性，需要对比分析不完全算法[26]，使用 CEC2017 基准测试函数的维度 30 将 UQSMA 与其不完全算法无迹西格玛点引导的黏菌算法 USMA、单纯使用拟反向学习改进的黏菌算法 QOSMA、单纯使用拟反射学习改进的黏菌算法 QRSMA 进行对比，表 5 展示了更加综合和简洁的 MAE 排名结果。由表 5 可见，UQSMA 算法排名第一，QOSMA 算法排名第二且优于原始 SMA 算法，USMA 和 QRSMA 排名第三第四且劣于原始 SMA 算法，说明了无迹西格玛点策略在单一使用时由于引入新的随机性会导致算法精度下降，固定持续的拟反向过程同样会破坏算法的搜索降低精度。但拟反射和拟反向与无迹西格玛点策略的组合使得算法表现最优，一方面综合的反向过程和适时的反向重构弥补了无迹西格玛点带来的随机性，提高了算法的稳定性；另一方面无迹西格玛点引入的新搜索行为可以帮助算法精进搜索细致性，弥补反向学习扩大搜索空间后不易捕捉到精进方向的盲目性，防止算法停滞。多种策略的结合相辅相成，既规避了单独策略使用时的缺陷，同时策略的组合使得策略发挥出更好的效果，有利于算法持续搜索效率和稳定性的提升。

表 5　MAE 不完全算法排名

Tab. 5　MAE incomplete algorithm ranking

算法	MAE	RANK
UQSMA	1.80E+05	1
QOSMA	2.07E+05	2
QRSMA	3.45E+05	3
USMA	3.64E+05	4

实验三：为了验证 UQSMA 算法在实际工程应用中的效果，使用 UQSMA 求解轿车侧面碰撞优化问题[27]，本问题主要考虑了 B 柱内径、B 柱加固、地板内径、横梁、门梁、门带加固、车顶纵梁的厚度、护栏高度、撞击位置等设计变量，相较于其他经典工程优化问题该问题维度更高、约束更为复杂。其目标函数、约束条件和决策变量取值范围如下所示。

$$f(X) = 1.98 + 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7$$

$$Subjected\ to:$$

$$g1(X) = 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} - 1 \le 0$$

$$g2(X) = 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0008757x_5x_{10} + 0.08045x_6x_9 + 0.00139x_8x_{11} + 0.00001575x_{10}x_{11} - 0.32 \le 0$$

$$g3(X) = 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 - 0.018x_2x_7 + 0.0208x3x8 + 0.121x_3x_9 - 0.00364x_5x_6 + 0.0007715x_5x_{10} - 0.000535x_6x_{10} + 0.00121x_8x_{11} - 0.32 \le 0$$

$$g4(X) = 0.074 - 0.061x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_9 + 0.227x_2^2 - 0.32 \le 0$$

$$g5(X) = 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10} - 32 \le 0$$

$$g6(X) = 33.86 + 2.95x_3 + 0.1792x_{10} - 5.057x_1x_2 - 11.0x_2x_8 - 0.0215x_5x_{10} - 9.98x_7x_8 + 22.0x_8x_9 - 32 \le 0$$

$$g7(X) = 46.36 - 9.9x_2 - 12.9x_1x_8 + 0.1107x_3x_{10} - 32 \le 0$$

$$g8(X) = 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} + 0.009325x_6x_{10} + 0.000191x_{11}^2 - 4 \le 0$$

$$g9(X) = 10.58 - 0.674x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} - 0.0198x_4x_{10} + 0.028x_6x_{10} - 9.9 \le 0$$

$$g10(X) = 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} - 0.000786x_{11}^2 - 15.7 \le 0$$

$$where,\ 0.5 \le x_1 - x_7 \le 1.5,\quad 0.192 \le x_8 - x_9 \le 0.345,\quad -30 \le x_{10} - x_{11} \le 30 \tag{19}$$

实验中种群规模设置为 30，最大迭代次数设置为 500，并独立运行 30 次计算得出统计结果。对比用算法及参数设置与实验一相同，由于边界控制方式不同 EGSWOA 算法没有使用。求解结果如表 6 所示，观察可知，UQSMA 算法虽在重复计算中最优值略逊于原始 SMA 算法及其改进算法，但多轮重复计算的均值优于所有对比用算法，且方差较小，说明了 UQSMA 在实际的优化问题中稳定性和鲁棒性更好。

表 6　轿车侧面碰撞问题优化结果

Tab. 6　The optimal result for car side impact design problem

	UQSMA	AFSMA	DOLSMA	SMA	HHO	WOA
决策变量						
x1	0.500000	0.500000	0.500540	0.500000	0.500000	0.500000
x2	1.125390	1.113507	1.121450	1.111922	1.136200	1.157687
x3	0.500000	0.500000	0.500646	0.500000	0.500000	0.500000
x4	1.288953	1.307631	1.293401	1.310424	1.499909	1.499801
x5	0.500000	0.500000	0.500059	0.500000	0.500000	0.500000
x6	1.499976	1.500000	1.500000	1.500000	1.499909	1.499801
x7	0.500079	0.500000	0.500000	0.500000	0.500000	0.547264
x8	0.344995	0.344933	0.345000	0.345000	0.343106	0.192000
x9	0.330035	0.207846	0.193399	0.305621	0.192000	0.192000
x10	-17.962397	-20.113941	-18.601569	-20.370221	-30.000000	-30.000000
x11	0.840285	-0.000497	-0.117660	-0.550441	-6.681273	5.181629
优化结果						
最优值	22.850268	22.845688	22.848868	22.846324	23.768084	24.040004
平均值	23.046099	23.211005	23.061354	23.096532	24.901598	27.681259
标准差	0.269430	0.355543	0.250977	0.276099	0.792983	1.701868
约束值						
g1(X)	-0.632695	-0.628194	-0.606523	-0.655046	-0.806886	-0.812774
g2(X)	-0.075343	-0.090981	-0.092269	-0.079360	-0.097477	-0.085725
g3(X)	-0.065285	-0.068270	-0.069034	-0.065982	-0.066378	-0.054259
g4(X)	-0.093732	-0.090221	-0.084601	-0.099203	-0.084642	-0.063955
g5(X)	-3.603556	-4.256299	-4.214778	-3.937901	-5.062355	-4.595731
g6(X)	-6.023978	-7.237602	-7.144709	-6.529494	-9.142410	-7.328384
g7(X)	-0.000799	-0.001838	-0.000935	-0.000773	-0.761912	0.000000
g8(X)	-0.000036	-0.000063	-0.000039	-0.000053	0.000000	-0.005395
g9(X)	-0.936811	-0.974801	-0.948969	-0.978981	-1.140159	-0.820629
g10(X)	-0.276582	-0.185096	-0.158891	-0.264329	-0.217056	-0.341223

4 结束语

针对 SMA 算法存在的不足，本文使用了拟反向和拟反射学习过程组成的综合反向和无迹西格玛点引导两个策略对原始 SMA 进行改进，提出了无迹西格玛点引导的拟反向黏菌算法 UQSMA，并将该算法应用于 CEC2017 测试函数进行求解。改进的反向学习框架将不同的反向生成过程按照特性融合于黏菌算法的勘探与开采过程中，扩大了原始黏菌算法的搜索空间，加速了勘探阶段的收敛速度，提高了算法的稳定性，对不同特点的测试函数具有普遍适用性；迭代过程中，使用无迹西格玛点引导黏菌算法种群的基本移动方式，使其移动方向更加逼近理论最优解，避免算法的停滞。此外，为了多维度综合地评估改进算法的有效性和同其他算法相比的竞争力，本文使用了 Wilcoxon 秩和检验、MAE 等指标，同时对轿车侧面碰撞的实际工程应用问题进行了求解和比较，多角度地证明了本文所提出算法的求解精度、鲁棒性和实用性。

在未来的研究中，希望对黏菌算法进行进一步的改进和测试，提高该算法针对更多样、更复杂问题的适用性，并将其应用于机器学习模型参数优化和投资组合问题求解等领域，进一步落实群智能算法的实际应用。

参考文献：

[1] Li Shimin, Chen Huiling, Wang Mingjing, *et al.* Slime mould algorithm: A new method for stochastic optimization [J]. Future Generation Computer Systems, 2020, 111: 300–323.

[2] Mostafa M, Rezk H, Aly M, *et al.* A new strategy based on slime mould algorithm to extract the optimal model parameters of solar PV panel [J]. Sustainable Energy Technologies and Assessments, 2020, 42 (November): 100849.

[3] Abdel-Basset M, Mohamed R, Chakrabortty R K, *et al.* An efficient binary slime mould algorithm integrated with a novel attacking-feeding strategy for feature selection [J]. Computers & Industrial Engineering, 2021, 153 (November 2020): 107078.

[4] Abdel-Basset M, Chang V, Mohamed R. HSMA _ WOA : A hybrid novel Slime mould algorithm with whale optimization algorithm for tackling the image segmentation problem of chest X-ray images [J]. Applied Soft Computing Journal, 2020, 95: 106642.

[5] Naik M K, Panda R, Abraham A. Normalized square difference based multilevel thresholding technique for multispectral images using leader slime mould algorithm [J]. Journal of King Saud University-Computer and Information Sciences, 2020.

[6] Rizk-Allah R M, Hassanien A E, Song D. Chaos-opposition-enhanced slime mould algorithm for minimizing the cost of energy for the wind turbines on high-altitude sites [J]. ISA transactions, 2022, 121: 191-205.

[7] 贾鹤鸣，刘宇翔，刘庆鑫，等. 融合随机反向学习的黏菌与算术混合优化算法 [J/OL]. 计算机科学与探索: 1-12 [2021-12-02]. http://kns. cnki. net/kcms/detail/11. 5602. TP. 20210714. 1817. 006. html. (Jia Heming, Liu Yuxiang, Liu Qingxin, *et al.* Hybrid algorithm of slime mould algorithm and arithmetic optimization algorithm based on random opposition-based learning [J/OL]. Journal of Frontiers of Computer Science and Technology: 1-12 [2021-12-02]. http://kns. cnki. net/kcms/detail/11. 5602. TP. 20210714. 1817. 006. html.)

[8] 郭雨鑫，刘升，张磊，等. 精英反向与二次插值改进的黏菌算法 [J/OL]. 计算机应用研究: 1-7 [2021-12-02]. https://doi. org/10. 19734/j. issn. 1001-3695. 2021. 02. 0175. (Guo Yuxin, Liu Sheng, Zhang Lei, *et al.* Elite opposition-based learning quadratic interpolation slime mould algorithm [J/OL]. Application Research of Computers: 1-7 [2021-12-02]. https://doi. org/10. 19734/j. issn. 1001-3695. 2021. 02. 0175.)

[9] 刘成汉，何庆. 改进交叉算子的自适应人工蜂群黏菌算法 [J/OL]. 小型微型计算机系统: 1-8 [2021-12-02]. http://kns. cnki. net/kcms/detail/21. 1106. TP. 20211114. 1427. 002. html. (Liu Chenghan, He Qing. Adaptive artificial bee colony slime mold algorithm with improved crossover operator [J/OL]. Journal of Chinese Computer Systems: 1-8 [2021-12-02]. http://kns. cnki. net/kcms/detail/21. 1106. TP. 20211114. 1427. 002. html.)

[10] Örnek B N, Aydemir S B, Düzenli T, *et al.* A novel version of slime mould algorithm for global optimization and real world engineering problems: Enhanced slime mould algorithm [J]. Mathematics and Computers in Simulation, 2022, 198: 253-288.

[11] Yu Caiyang, Heidari A A, Xue Xiao, *et al.* Boosting quantum rotation gate embedded slime mould algorithm [J]. Expert Systems with Applications, 2021, 181: 115082.

[12] Abualigah L, Diabat A, Mirjalili S, *et al.* The arithmetic optimization algorithm [J]. Computer methods in applied mechanics and engineering, 2021, 376: 113609.

[13] Tizhoosh H R. Opposition-based learning: a new scheme for machine intelligence [C]// International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06) . IEEE, 2005, 1: 695-701.

[14] Rahnamayan S, Tizhoosh H R, Salama M M A. Quasi-oppositional differential evolution [C]// 2007 IEEE congress on evolutionary computation. IEEE, 2007: 2229-2236.

[15] Ewees A A, Abd Elaziz M, Houssein E H. Improved grasshopper optimization algorithm using opposition-based learning [J]. Expert Systems with Applications, 2018, 112: 156-172.

[16] Fan Qian, Chen Zhenjian, Xia Zhanghua. A novel quasi-reflected Harris hawks optimization algorithm for global optimization problems [J]. Soft Computing, 2020, 24 (19): 14825-14843.

[17] Wineberg M, Oppacher F. The underlying similarity of diversity measures used in evolutionary computation [C]// Genetic and evolutionary computation conference. Springer, Berlin, Heidelberg, 2003: 1493-1504.

[18] Solis F J, Wets J B. Minimization by Random Search Techniques [J]. Mathematics of Operations Research, 1981, 6 (1): 19-30.

[19] Choi T J, Togelius J, Cheong Y G. A fast and efficient stochastic opposition-based learning for differential evolution in numerical optimization [J]. Swarm and Evolutionary Computation, 2021, 60: 100768.

[20] Guo Wenyan, Xu Peng, Dai Fengqun, *et al.* Improved harris hawks optimization algorithm based on random unscented sigma point mutation strategy [J]. Applied Soft Computing, 2021: 108012.

[21] Mirjalili S, Lewis A. The whale optimization algorithm [J]. Advances in engineering software, 2016, 95: 51-67.

[22] 肖子雅，刘升. 精英反向黄金正弦鲸鱼算法及其工程优化研究 [J]. 电子学报, 2019, 47 (10): 2177-2186. (Xiao Ziya, Liu Sheng. Study on elite opposition-based golden-sine whale optimization algorithm and its application of project optimization [J]. Acta Electronica Sinica, 2019, 47 (10): 2177-2186.)

[23] Heidari A A, Mirjalili S, Faris H, *et al.* Harris hawks optimization: Algorithm and applications [J]. Future generation computer systems, 2019, 97: 849-872.

[24] Derrac J, García S, Molina D, *et al.* A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms [J]. Swarm and Evolutionary Computation, 2011, 1 (1): 3-18.

[25] 李守玉，何庆，杜逆索. 混沌反馈共享和群体协同效应的蝴蝶优化算法 [J/OL]. 计算机科学与探索: 1-12 [2021-05-19]. http://kns. cnki. net/kcms/detail/11. 5602. TP. 20210128. 1109. 014. html. (Li Shouyu, He Qing, Du Nisuo. Butterfly optimization algorithm for chaotic feedback sharing and group synergy [J/OL]. Journal of Ftontiers of Computer Science and Technology: 1-12 [2021-05-19]. http://kns. cnki. net/kcms/detail/11. 5602. TP. 20210128. 1109. 014. html.)

[26] 张新明，姜云，刘尚旺，等. 灰狼与郊狼混合优化算法及其聚类优化 [J/OL]. 自动化学报: 1-17 [2021-05-19]. https://doi. org/10. 16383/j. aas. c190617. (Zhang Xinmin, Jiang Yun, Liu Shangwang, *et al*. Hybrid coyote optimization algorithm with grey wolf optimizer and its application to clustering optimization [J/OL]. Acta Automat. Sin. : 1-17 [2021-05-19]. https://doi. org/10. 16383/j. aas. c190617.)

[27] Youn B D, Choi K K, Yang R J, *et al.* Reliability-based design optimization for crashworthiness of vehicle side impact [J]. Structural and Multidisciplinary Optimization, 2004, 26 (3): 272-283.